LITERATURE REVIEW:
Explore how the OpenMP can improve application performance

Dehui Yu
School of Computer Science
University of Ottawa
Ottawa, Canada K1N 6N5
dyu105@uottawa.ca

October 7, 2022

1 Introduction

Parallel computing has been around for many years, but recently interest in the high performance computing community has begun to grow. This is because hardware is getting cheaper and cheaper.In my project, I want to explore the use of OpenMP.OpenMP (Open Multi-Processing) is an application programming interface (API) that supports multi-platform shared-memory multiprocessing programming in C, C++, and Fortran,on many platforms, instruction-set architectures and operating systems, including Solaris, AIX, FreeBSD, HP-UX, Linux, macOS, and Windows.[3][2]

Our project wants to use Monte Carlo algorithms to determine the effectiveness of OpenMP in running speed. What is Monte Carlo method? When the problem to be solved is the probability of a certain random event or the expected value of a certain random variable, the probability of this random event is estimated by the frequency of this event through some "experimental" method, or some numerical characteristics of this random variable are obtained and taken as the solution of the problem. Usually, Monte Carlo methods solve various mathematical problems by constructing random numbers that conform to certain rules. The Monte Carlo method is an effective way to find numerical solutions for problems that are too complex to obtain analytical solutions or for which no analytical solutions are available at all. The most common application of the general Monte Carlo method in mathematics is the Monte Carlo integral, which can be used to calculate PI.[4][6]

Case study 1: Calcualte PI using Monte Carlo algorithms.

The more the number of points scattered, the more accurate the PI will be. However, there will be a problem at this time, that is, after too many points, the algorithm will take a long time to run. And exactly the simulation of the scatter point this thing can be implemented in parallel, that is to say, the scatter point does not affect, so you can let multiple processes or threads scatter points at the same time, and finally all into/thread scatter points together statistics, to get a higher accuracy of PI.[10]

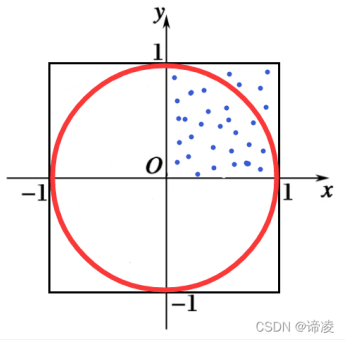

Figure 1: Calculating PI using Monte Carlo algorithms[10]

Case study 2: Matrix Multiplication using OpenMP

In addition, our project also explores complicated matrix multiplication using OpenMP. Matrix multiplication is a basic tool of linear algebra. If A and B are matrices, then the coefficients of the matrix C=AB are equal to the dot product of rows of A with columns of B. The naive matrix multiplication algorithm has a computational complexity of O(n^3). [1] In our project, we aim to use OpenMP to minimize the computational complexity based on parallel computation.

2 Literature Review

Since our project mainly focus on utilizing OpenMP to improve application performance, our first step was to investigate why and how to use OpenMP. This article[2] tells us why we should run our code in parallel, and the answer is obvious: to reduce the time it takes to run. This article introduces an interesting point that simply adding more processing won't significantly improve the runtime of your code. Figure 2 shows the scalability of parallelised program. It clearly shows that simply adding more processors will not always make a significant improvement to the run time of a code. Therefore, in our project, we will also explore the optimal Scalability of Parallelisation. At the same time, it showed me that OpenMP works in C, C++, and Fortran, which allowed us to confirm the language we will use in our project.

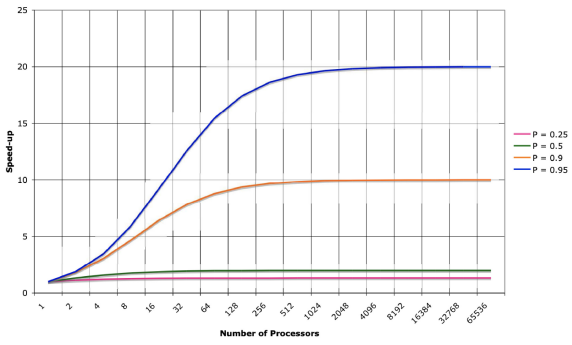

Figure 2: Scalability of parallelised program[2]

Before reading this paper[5], we didn't know the difference between OpenMP and MPI. OpenMP and MPI are two approaches for parallel programming, and the following Figure 3 illustrates the difference. As can be seen from the Figure 3, MPI is designed for distributed, while OpenMP is designed for shared memory. Since there are two ways to implement parallel programming, this also leads us to also compare OpenMP and MPI methods in our project, we proposed the research question: which one is better on exact same program and number of CPUs?

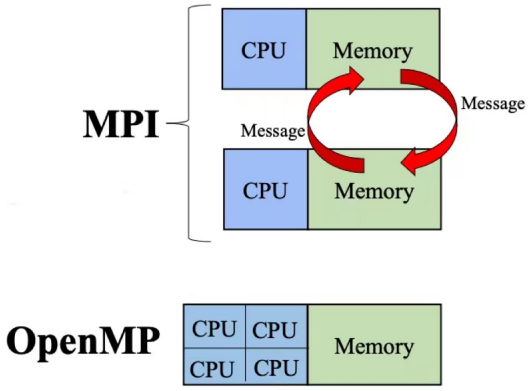

Figure 3: OpenMP vs MPI [5]

In our case study 2, we are inspired by article[1]. Their study compares the speed of complicated parallel matrix computation using OpenMP under different number of CPUs. Our project will reproduce their experimental results, it is worth noting that we utilize block matrix to optimize the performance. Blocked matrix multiplication is a technique in which you separate a matrix into different 'blocks' in which you calculate each block one at a time.[11] This can be useful for larger matrices where spacial caching may come into play. In our project we will be doing blocked matrix multiplication in a parallel fashion, in which each element of the block is calculated on a thread.

Finally, OpenMP has a lot of practical uses, which means OpenMP has great potential on performance improvement, and that's the reason why we want to do projects related to OpenMP. Such as: Paper[9] Disposing an Optimal Strategy to Parallelize the Solution to Large 3D Magneto-quasi-static (MQS) Problems by combining the MPI and OpenMP approaches.Article[7] presents a novel hybrid graph coloring algorithm and discuss how to obtain the best performance on such systems from algorithmic, system and engineering perspectives by using OpenMP. Paper [8] apply openmp to the Image Warping,on a 512 $\times$ 512 image, and using a quad-core 2.4 GHz CPU, the single-threaded code requires 62.2 ms to process the image, while the multithreaded code requires 17.7 ms, corresponding to a 3.5$\times$ speedup. For the most part, these papers show a speed boost when using OpenMP, but many of them don't talk about what happens when you increase the number of parallel computing nodes infinitely. This is one of the highlights of my project because I will take care of it.

3 Summary

To sum up, my project is to explore the effectiveness of parallel programming based on OpenMP and MPI compared with serial programming. We foucs on two case studies:

(1)Calculate PI using Monte Carlo algorithms; (2)Matrix Multiplication using OpenMP. And our project has three clear goals:

1. Conduct a detailed literature review to investigate why OpenMP is now an industry standard API for shared-memory programming and its real-world applications.
2. Try to use OpenMP for scientific calculations (matrix multiplication, PI calculation based on Monte Carlo algorithm) to analyze application performance by ourself.
3. Investigate the connection and differences between OpenMP and MPI, and also try to use MPI API in our application, then compare performance of MPI and OpenMP.

References

[1] Haitham A Alasha'ary, Khaled M Matrouk, Abdullah I Al-Hasanat, Ziad A Alqadi, and Hasan M Al-Shalabi. Improving matrix multiplication using parallel computing. *International Journal on*, page 346, 2013.

[2] Rohit Chandra, Leo Dagum, David Kohr, Ramesh Menon, Dror Maydan, and Jeff McDonald. *Parallel programming in OpenMP*. Morgan kaufmann, 2001.

[3] Leonardo Dagum and Ramesh Menon. Openmp: an industry standard api for shared-memory programming. *IEEE computational science and engineering*, 5(1):46–55, 1998.

[4] Robert L Harrison. Introduction to monte carlo simulation. In *AIP conference proceedings*, volume 1204, pages 17–21. American Institute of Physics, 2010.

[5] Haoqiang Jin, Dennis Jespersen, Piyush Mehrotra, Rupak Biswas, Lei Huang, and Barbara Chapman. High performance computing using mpi and openmp on multi-core parallel systems. *Parallel Computing*, 37(9):562–575, 2011.

[6] Dirk P Kroese, Tim Brereton, Thomas Taimre, and Zdravko I Botev. Why the monte carlo method is so important today. *Wiley Interdisciplinary Reviews: Computational Statistics*, 6(6):386–392, 2014.

[7] Ahmet Erdem Sariyüce, Erik Saule, and Ümit V Çatalyürek. Scalable hybrid implementation of graph coloring using mpi and openmp. In *2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum*, pages 1744–1753. IEEE, 2012.

[8] Greg Slabaugh, Richard Boyes, and Xiaoyun Yang. Multicore image processing with openmp [applications corner]. *IEEE Signal Processing Magazine*, 27(2):134–138, 2010.

[9] Salvatore Ventre, Francesca Cau, Andrea Chiariello, Gaspare Giovinco, Antonio Maffucci, and Fabio Villone. Fast and accurate solution of integral formulations of large mqs problems based on hybrid openmp–mpi parallelization. *Applied Sciences*, 12(2):627, 2022.

[10] Timothy Williamson. Calculating pi using the monte carlo method. *The Physics Teacher*, 51(8):468–469, 2013.

[11] Fuzhen Zhang. Block matrix techniques. In *The Schur complement and its applications*, pages 83–110. Springer, 2005.